PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30	A1	(11) International Publication Number: WO 98/12650 (43) International Publication Date: 26 March 1998 (26.03.98)

(21) International Application Number: PCT/US96/14994

(22) International Filing Date: 18 September 1996 (18.09.96)

(71) Applicant: MCI COMMUNICATIONS CORPORATION [US/US]; 1133 19th Street, N.W., Washington, DC 20036 (US).

(72) Inventor: DeVRIES, Larry, R.; 500 Second Street, S.E., Cedar Rapids, IO 52401 (US).

(74) Agent: WOO, Louis; Pollock, Vande Sande & Priddy, P.O. Box 19088, Washington, DC 20036 (US).

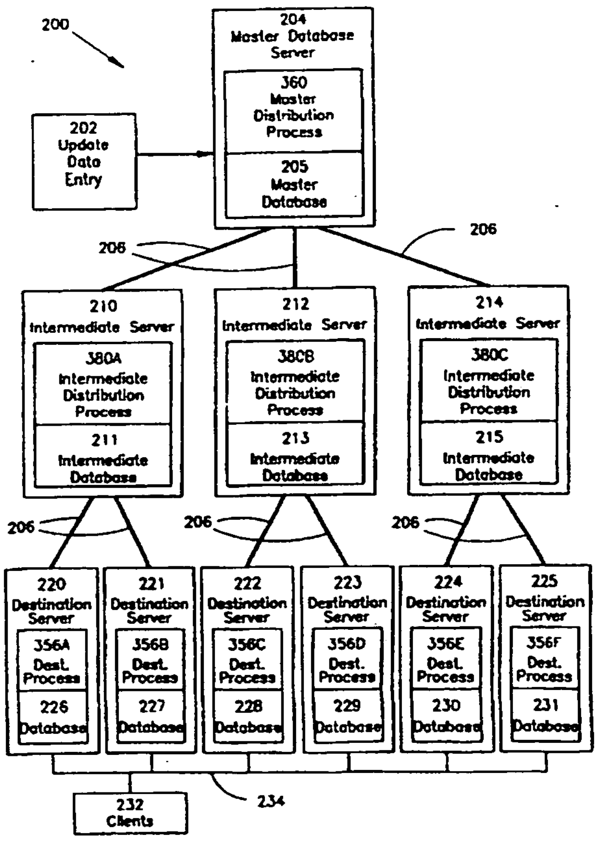
(81) Designated States: CA, JP, MX, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM AND METHOD FOR HIERARCHICAL DATA DISTRIBUTION

(57) Abstract

The hierarchical data distribution system includes a top level master database system, bottom level client servers, each with its own database copy, and at least one intermediate database level. The entry of update data into the system invokes the distribution process. First, the master database system is updated. Then, the master database system updates several database systems at the first intermediate database level. Each database system at the first intermediate database level then updates several database systems at the next lower database level. This process continues until the lowest level database systems, the client servers, have been updated. The distribution process performs any necessary reformating, data assembly and data view processing before transmitting the update data.



SYSTEM AND METHOD FOR HIERARCHICAL DATA DISTRIBUTION

BACKGROUND OF THE INVENTION

Field Of The Invention

The invention relates generally to distributed databases, and in particular, to hierarchical distribution of data to distributed databases.

Background Information

Local and wide area networks allow multiple clients to simultaneously access databases stored on a server. As the number of clients grows, database accesses increase. If the database is stored on only one server, the maximum throughput of that server is a limit on the amount of access clients have to the database. One solution to this problem is to store a complete copy of the database on multiple servers and distribute client accesses relatively evenly among the servers. This allows database access far greater than would be possible with only one server.

A problem arises, however. When a database which is stored on multiple servers must be updated, the copy on each server must be updated. This can be accomplished by updating a master copy of the database, then updating the other copies from the master copy. During the update process, not all copies of the database are identical. This is because at any given point in the process, some copies have been updated and some have not. This condition is known as transitional inconsistency. Some clients accessing the database will obtain updated data and some will not. A problem arises if the period of transitional inconsistency becomes so long as to seriously affect a significant number of clients. If the number of database copies is large, it will take an unacceptably long time for the master to update all the copies. A need exists to update multiple database copies with improved performance.

A typical prior art system is exemplified by United States Patent 5,251,094 to Everson et al. In this system, a first program in a collector node instructs a second program in a collectee node to send all updates to a database since the last conversation. The second program processes queries to retrieve any changes made since the last conversation between the collector and collectee nodes and sends the data to the first program, which updates the copy of the database on its own system. This is an example of a puller type database system, that is, a system in which the database server which is to be updated initiates a request for updating and communicates the request to the master database. The master database then transmits the requested data to the database server.

A problem arises with puller type systems when the communications between the master database and a server are not functioning. The lack of communications prevents the request for update from the server from reaching the master. Because the master only sends updates when requested, the server will never be updated. Furthermore, the master will not soon detect that the server is out of communication because the absence of requests from the server is the normal condition. A need exists to detect when communication cannot be established with a server containing a database to be updated

A further problem arises when the update process is interrupted during an update communication. Such an interruption may occur, for example, due to a communications outage or due to failure of the master database system or the server. Interruption of the update process may cause the replicated data to be missing or defective. A need exists for guaranteed delivery of update data, in spite of interruptions in the update process.

Another problem arises with puller type systems because of the periodic nature of the update requests. If the period between update requests is set too long, transitional inconsistency will likewise be of long duration. If the period between update requests is set too short, network resources are wasted. Many update requests are sent which do not result in update data being transmitted simply because not enough time has elapsed to allow any update data to be entered. A need exists to reduce the duration of transitional inconsistency present during updates of multiple database copies, and at the same time reduce waste of network resources.

Another type of problem arises when the databases on different servers are in different formats. The change to the master database cannot simply be replicated into the destination databases. The change must be reformatted and entered into the destination database in its specified format. A need exists to reformat database updates to the format specified by the database to be updated.

SUMMARY OF THE INVENTION

The hierarchical data distribution system (HDDS) of the present invention provides a system and method for updating multiple database copies with improved performance. HDDS reduces the duration of transitional inconsistency and the waste of network resources during updates of multiple database copies. HDDS provides an indication when communication cannot be established with a server containing a database to be updated. HDDS provides guaranteed delivery of update data, in spite of interruptions in the update process. HDDS also provides reformatting of database updates, as well as more sophisticated data assembly and data view processing of database updates.

HDDS includes a top level master database system, bottom level client servers, each with its own database copy, and at least one intermediate database level. First, the master database system is updated. Then, the master database system updates several database systems at the first intermediate database level. Each database system at the first intermediate database level then updates several database systems at the next lower database level. This may be a lower level intermediate level or it may be client servers. This process continues until the lowest level database systems, the client servers, have been updated. Each higher level database system must update fewer lower level servers and overall update performance is improved.

HDDS is an event driven pusher type system. The entry of update data into the system invokes the distribution process. If communications to a destination server are not functional, the distribution system detects this immediately because it is unable to establish communications with the destination server. Transitional inconsistency is reduced because the distribution process is invoked for all destination servers at the same time. System resources are not wasted because communications are only established when there is data to be updated. In addition, the distribution process performs any necessary reformatting, data assembly and data view processing before transmitting the update data.

An example of a data record that would be handled by the HDDS system is a customer subscription entry for a calling card. The record would contain various information about the customer such as the Card Number, the customer name, the business name, card restrictions, available feature sets, etc. Another example of a data record is a personal 800 number subscription entry. This record would contain the 800 number, the Personal Identification Number (PIN), and the terminating number.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a prior art network data distribution system 100.

Fig. 2a is a block diagram of an exemplary network data distribution system 200, in accordance with the present invention.

Fig. 2b is an exemplary block diagram of a master or intermediate database server of Fig. 2a.

Fig. 2c is an exemplary block diagram of a destination database server of Fig. 2a.

Fig. 3a is a diagram of data flow in the exemplary network data distribution system 200 of Fig. 2a.

Fig. 3b is a block diagram of processing implemented in network data distribution system 200 of Fig. 2a.

Fig. 3c is a diagram of data flow in an exemplary network in which there are two levels of intermediate distribution processing, in accordance with the present invention.

Fig. 4a is a flow diagram of a master distribution process 360 implemented in master server 204 of Fig. 2a.

Fig. 4b is a flow diagram of a subprocess of step 418 of master distribution process 360, implemented in master database server 204 of Fig. 2a.

Fig. 5a is a flow diagram of an intermediate distribution process 380 implemented in each intermediate server of Fig. 2a.

Fig. 5b is a flow diagram of a subprocess of step 518 of intermediate distribution process 380, implemented in each intermediate server of Fig. 2a.

Fig. 5c is a flow diagram of a subprocess of step 502 of intermediate distribution process 380, implemented in each intermediate server of Fig. 2a.

Fig. 6a is a flow diagram of a destination process 356 implemented in each destination server of Fig. 2a.

Fig. 6b is a flow diagram of a subprocess of step 602 of destination process 356. implemented in each destination server of Fig. 2a.

Fig. 7 is an exemplary format of an update data record used in system 200.

DETAILED DESCRIPTION OF THE INVENTION

Fig. 1 is a block diagram of a prior art network data distribution system 100. Included are update data entry 102. which represents one or more terminals used to enter data to update the database. Update data entry 102 is connected to master database server 104. Master server 104 contains the master copy 105 of the database being updated. Updates entered on update data entry 102 are entered into master database 105. Master database server 104 is connected to network 110. Network 110 also connects to multiple destination database servers 106 - 108. Updates made to master database 105 are communicated by master database server 104 over network 110 to each destination database server 106 - 108. Each destination database server 106 - 108 is also connected to network 114. Network

114 allows clients 112 to access the databases on destination database servers 106 - 108.

Fig. 2a is a block diagram of an exemplary network data distribution system 200, in accordance with the present invention. Included are update data entry 202, which represents one or more terminals used to enter data to update the database. Update data entry 202 is connected to master database server 204. Master server 204 is, in one embodiment, a mainframe computer system, such as, for example, an ES/9000. Master server 204 contains the master copy 205 of the database being updated. Master server 204 also executes master distribution process 360. Updates entered on update data entry 202 are promptly entered into master database 205. Master database server 204 is connected to network 206. Network 206 is a standard network such as SNA.

Network 206 couples master server 204 to multiple intermediate database servers 210, 212 and 214. Servers 210, 212 and 214 are intermediate in level between master server 204 and destination database servers 220 - 225. Although only one intermediate level is shown in Fig. 2a, it is also in accordance with the present invention to have more than one intermediate level. In one embodiment, each intermediate database server is a computer system such as, for example an RS/6000. Intermediate database servers 210, 212 and 214 contain intermediate copies 211, 213 and 215 of the database being updated. Each intermediate database server 210, 212 and 214 also executes an instance of intermediate distribution process 380, represented by blocks 380A, 380B and 380C. Although three intermediate database servers 210, 212 and 214 are shown, these represent multiple intermediate database servers that may be connected by network 206. Network 206 also couples each intermediate database server to multiple destination database servers 220 - 225. Each destination database server is a computer system such as, for example, an RS/6000. Destination database servers 220 - 225 contain lowest level copies 226 - 231 of the database being updated. Each destination database server also executes an instance of destination process 356, represented by blocks 356A-F. Although six destination database servers are shown, these represent multiple destination database servers that may be connected to network 206. Clients 232, which access the destination database servers, are coupled to the destination database servers by local/wide area network 234 which is a standard network such as, for example, Ethernet or Token Ring.

Updates made to master database 205 are communicated from master database server 204 over network 206 to intermediate database servers 210, 212 and 214. These updates are then communicated from the intermediate database servers to the destination database servers. The network connections 206 shown in Fig. 2a are exemplary of Systems Network Architecture (SNA). Such a network would typically utilize Advanced Program to Program Communications (APPC) protocol and LU6.2 devices. However, any network architecture which provides the necessary communication connectivity may be used.

Fig. 2b is an exemplary block diagram of a database server 250, which is representative of master server 204 and intermediate servers 210, 212 and 214 of Fig. 2a. In one embodiment, each database server is a computer system such as, for example an ES/9000 used for the master server or a RS/6000 used for an intermediate server. System 250 includes a CPU 252, for executing program instructions and processing data, memory 260, for storing program instructions executed by and data processed by CPU 252, and at least one I/O adapter 254, for communicating with other devices and transferring data in and out of the computer system, for example, over network 206. System 250 may also may include an operator interface 256, for providing status information to and accepting commands from a system operator. All these elements are interconnected by bus 251, which allows data to be intercommunicated between the elements.

Memory 260 may include both volatile storage devices, such as, for example, random access memory (RAM) and non-volatile storage devices, such as, for example, hard disk, floppy disk, optical disk, electrically-alterable programmable read only memory, battery-backed up RAM, etc. Memory 260 is accessible to CPU 252 over bus 251 and includes operating system 269, process 261, which is either master distribution process 360 or intermediate distribution process 380, input queue 265, output queues 266, distribution rules tables 267, data assembly routines 268, database 270 and data view routines 271. Process 261 includes receiver process 262, distribution process 263 and communications processes 264. These processes are similar in both master distribution process 360 and intermediate distribution process 380. CPU 252 execution of receiver process 262 receives data and stores it in input queue 265. In the case of master server 204, data is received from update data entry 202. In the case of intermediate level servers, data is received from higher level servers, either master server 204 or higher level intermediate servers. CPU 252 execution of distribution process 263 causes process 263 to remove data from input queue 265, process it with distribution rules table 267, data assembly routines 268 and data view routines 271 and store the result in one of the output queues 266. CPU 252 execution of communications processes 264 causes each instance of process 264 to remove data from its respective output queue and transmit it to lower level receivers. In the case of the master server, the lower level receivers are intermediate level servers. In the case of an intermediate level server, the lower level receivers are either lower level intermediate servers or destination database servers. Database 270 is a master or intermediate copy of each database to be updated.

Fig. 2c is an exemplary block diagram of a database server 280, which is representative of destination servers 220-225 of Fig. 2a. Each database server is a computer system such as, for example a RS/6000. System 280 includes a CPU 282, for executing program instructions and processing data, memory 288, for storing program instructions executed by and data processed by CPU 282, and at least one I/O adapter 284, for communicating with other devices and transferring data in and out of the computer system, for example, over network 206. System 280 may also may include an operator interface 286, for providing status information to and accepting commands from a system operator. All these elements are interconnected by bus 281, which allows data to be intercommunicated between the elements.

Memory 288 may include both volatile storage devices, such as, for example, random access memory (RAM) and non-volatile storage devices, such as, for example, hard disk, floppy disk, optical disk, electrically-alterable programmable read only memory, battery-backed up RAM, etc. Memory 288 is accessible to CPU 282 over bus 281 and includes operating system 295, database 294 and destination process 356, which includes receiver process 291, update process 292 and input queue 293. CPU 282 execution of receiver process 291 receives data from an intermediate server and stores it in input queue 293. CPU 252 execution of update process 292 causes process 292 to remove data from input queue 293 and enter it into database 294.

Fig. 3a is a diagram of data flow in the exemplary network data distribution system 200 of Fig. 2a. Included are update data entry 302, which represents one or more terminals used to enter data to update the database. Update data entry 302 communicates the entered data to data input process 304. Data input process 304 supports the input of both customer and configuration data into the database. Data input process 304 communicates the input data to master distribution process 360. Master distribution process 360, implemented on master server 204, processes the incoming data using soft-coded distribution rules, data assembly and data view routines. Master distribution process 360 then distributes the data to multiple instances of intermediate distribution process 380, represented by intermediate distribution processes 380A, B and C. Intermediate distribution process 380A distributes the data sent from master distribution process 360 to the instances of destination process 356 represented by destination processes 356A and B. Intermediate distribution process 380B distributes the data sent from master distribution process 360 to the instances of destination process 356 represented by destination processes 356C and D. Intermediate distribution process 380C distributes the data sent from master distribution process 360 to the instances of destination process 356 represented by destination processes 356E and F.

Fig. 3b is a block diagram of processing implemented in network data distribution system 200 of Fig. 2a. Data entry 302 transmits an update data record to data input process 304. Data input process 304 communicates the update data record to receiver process 361 of master distribution process 360. Receiver process

361 receives the update data record, stores it to master database 205 and writes it to input queue 362. The presence of data in queue 362 causes distribution process 364 to be invoked. Distribution process 364 processes the update data record using distribution rules, data assembly and data view routines 366. This produces what is termed a distribution record. Distribution process 364 then writes the distribution record to one or more communication output queues 368 to 370 specified by the distribution rules. Each update data record is processed separately for each output queue to which it is to be written. Therefore, an update data record written to an output queue may be processed differently from the same record written to a different output queue. Queues 368 to 370 represent multiple output queues which exist in master distribution process 360. The presence of data in a queue 368 to 370 causes a corresponding communication process 372 to 374 to be invoked. Communication processes 372 to 374 represent multiple communication processes which may be invoked in master distribution process 360. Each communication process 372 to 374 communicates with and transmits data to a different intermediate distribution process.

Receiver process 381 of intermediate distribution process 380 receives the distribution record, stores it to the intermediate database, for example 211, and writes it to its input queue 382. Although for clarity only one intermediate distribution process 380 is shown, there is actually an intermediate distribution process 380 corresponding to each communication process 372 - 374 of master distribution process 360. In one embodiment each intermediate server executes only one intermediate distribution process 380. In another embodiment, some intermediate servers may execute more than one intermediate distribution process.

This is useful, for example, because it allows the effects on the overall system of non-operational hardware to be minimized. The intermediate distribution processes which would otherwise run on the non-operational hardware may be run on the remaining operational hardware. This may achieved using standard multi-tasking, multi-threaded operating systems. The presence of data in input queue 382 causes distribution process 384 to be invoked. Distribution process 384 processes the distribution record using distribution rules, data assembly and data view routines 386. Distribution process 384 then writes the processed distribution record to one or more communication output queues 388 to 390 specified by the distribution rules. Each update data record is processed separately for each output queue to which it is to be written. Therefore, an update data record written to an output queue may be processed differently from the same record written to a different output queue. Queues 388 to 390 represent multiple output queues which exist in intermediate distribution process 380. The presence of data in a queue 388 to 390 causes a corresponding communication process 392 to 394 to be invoked. Communication processes 392 to 394 represent multiple communication processes which may be invoked in intermediate distribution process 380. Each communication process 372 to 374 communicates with and transmits data to a different downline process. Each downline process may either be a destination process 356 or another intermediate distribution process 380'. If the downline process is destination process 356, destination process 356 receives the record and stores it in its database 358. If the downline process is lower-level intermediate distribution process 380', process 380' processes the record and communicates it in turn to a downline process which may likewise be either a destination process 356 or another lower-level intermediate distribution process 380''. As a result, there may be multiple levels of intermediate distribution processing between master distribution process 360 and destination process 356.

Fig. 3c is a data flow diagram of an exemplary network in which there are two levels of intermediate distribution processing. Master distribution process 360 processes the incoming data and distributes it to multiple instances of intermediate distribution process 380, represented by intermediate distribution processes 380A, B and C. Intermediate distribution processes 380A, B and C in turn process the data and distribute it to multiple instances of intermediate distribution process 380 at a lower intermediate level of hierarchy. The multiple instances of intermediate distribution process 380 are represented by intermediate distribution processes 380D - I. Intermediate distribution processes 380D - I then process the data and distribute it to multiple instances of destination process 356 represented by destination processes
356A - Z.

Fig. 4a is a flow diagram of master distribution process 360, which is implemented by program instructions executed by the CPU of master server 204. It is best understood when viewed in conjunction with Fig. 2a. Process 360 begins with step 402, in which an update data entry transaction is received by the receiver process of master database server 204. In step 404, the update information is written by the receiver process of server 204 to master database 205. In step 406, the update data record is written to the input queue. In step 408, the distribution process is invoked. The distribution process is event driven, that is, the writing of a record to the input queue causes the distribution process to be invoked. In step

410, the distribution process determines the data type of the update data record being processed. In step 412, the update data record is processed as specified by the distribution rules for that record's data type. The distribution rules specify reformatting and transformations to be performed to the update data record, as well as more extensive data assembly and data view routines. The data assembly routines are capable of assembling an output update data record from one or more received update data records and from records already present in the database. The data view routines are capable of individually processing the fields of one or more received update data records and other records and selectively including these fields in any arrangement in the output update data record. The rules also specify the lower level servers and output queues to which each update data record is to be written. Each update data record is processed separately for each output queue to which it is to be written. Therefore, an update data record written to an output queue may be processed differently from the same record written to a different output queue. In step 414, the update data record is written to each output queue specified by the distribution rules. In step 416, the communication processes are invoked. The communication processes are event driven, the writing of a record to an output queue causes the corresponding communication process to be invoked. In step 418, each active communication process transmits its associated output queue to the specified receivers. Process 360 then ends.

Fig. 4b is a flow diagram of a subprocess of step 418 of master distribution process 360. Subprocess 418 interacts with the subprocess of step 502 of Fig. 5a, as is explained below. Subprocess 418 is entered from step 416 of master distribution process 360 and begins with step 418-1 in which the master server logs its state and backs up the data to be communicated. In step 418-2, the master server attempts to establish communication with a lower level server. If the communication attempt is successful, the process goes to step 418-3, in which the master server attempts to transmit the update to the lower level server. If the transmission attempt is successful, the process goes to step 418-4, in which the master server waits to receive an acknowledgment that the transmitted data was successfully received and saved in safe storage on the lower level server. If the acknowledgment is successfully received, the process ends.

If any of steps 418-2, 418-3, or 418-4 fail, the process goes to step 418-5, in which it is determined whether the master server has failed. If the master server has not failed, the process goes to step 418-6, in which the cause of the failure of steps 418-2, 418-3, or 418-4, and the correct retry action, are determined. Step 418-2 will fail if communications cannot be established with the lower level server, for example, if there is a communications outage or if the lower level server has failed. Step 418-3 will fail if the update cannot be transmitted to the lower level server, typically for similar reasons. Step 418-4 will fail if the master server fails to receive an acknowledgment, for example, if a timeout occurs before the acknowledgment is received or a non-acknowledgment is received. Again, typical causes are a communications outage or failure of the lower level server. In step 418-7, the alarm counters are updated depending upon the failure cause determined in step 418-6. In step 418-8, if any of the alarm counters have exceeded their specified values, an alarm is transmitted to the network alarm reporting system. The specified values of the alarm counters are softcoded and can readily be changed. The alarm provides a timely indication of an update failure and may be used to indicate to the network operator that corrective action should be taken on the lower level server. The process then continues with either step 418-2, 418-3, or 418-4 depending on the cause of the failure and the retry action determined in step 418-6.

If the master server has failed, the master server is not operating. Typically, failure of the master server will cause alarms to be transmitted to the network alarm reporting system independently of this process. The process goes to step 418-9, which is a block that simply indicates that the process waits until operation of the master server is restored. Once the master server is again operational, the process goes to step 418-10, in which the state previously logged and the data previously backed up are restored. The process then goes to step 418-2 and resumes the delivery attempt. The process continues indefinitely until transmission is successful or until the process is stopped by external intervention. for example, in response to alarms reported by the network alarm reporting system. When transmission is successful. the process ends. In this way, HDDS provides guaranteed delivery of all updates. as well as timely indication of failure of a server or a communication outage.

Fig. 5a is a flow diagram of intermediate distribution process 380. which is implemented by program instructions executed by the CPU of each intermediate server. It is best understood when viewed in conjunction with Fig. 2a. Process 380 begins with step 502, in which an update data record is received by the receiver process of an intermediate server from a higher level system, which is either master database server 204 or a higher level intermediate system. In step 504, the update data record is written by the receiver process of the intermediate server to the intermediate database. In step 506, the update data record is written to the distribution queue. In step 508, the distribution process is invoked. The distribution process is event driven, that is, the writing of a record to the distribution queue causes the distribution process to be invoked. In step 510, the distribution process determines the data type of the update data record being processed. In step 512, the update data record is processed as specified by the distribution rules for that record's data type. The distribution rules specify reformatting and transformations to be performed to the update data record, as well as more extensive data assembly and data view routines. The data assembly routines are capable of assembling an update data record from one or more received update data records and from records already present in the database. The data view processing routines are capable of individually processing the fields of one or more received update data records and other records and selectively including these fields in any arrangement in the output update data record. The rules also specify the lower level servers and output queues to which each update data record is to be written. Each update data record is processed separately for each output queue to which it is to be written. Therefore, an update data record written to an output queue may be processed differently from the same record written to a different output queue. In step 514, the update data record is written to each output queue specified by the distribution rules. In step 516, the communication processes are invoked. The communication processes are event driven, the writing of a record to an output queue causes the corresponding communication process to be invoked. In step 518, each active communication process transmits its associated output queue to the specified receivers. Process 380 then ends.

Fig. 5b is a flow diagram of a subprocess of step 518 of intermediate distribution process 380. The subprocess of step 518 of intermediate distribution process 380 is similar to step 418 of master distribution process 360, except that the transmitting server is an intermediate server and the receiving server is either an intermediate server or a destination server. Subprocess 518 interacts with the subprocess of step 502, executing on lower-level intermediate servers, and with the subprocess of step 602, executing on destination servers, as is explained below.

Fig. 5c is a flow diagram of a subprocess of step 502 of intermediate distribution process 380. The subprocess 502 interacts with the subprocess of step 418 of master distribution process 360 and with the subprocess of step 518 of instances of intermediate distribution process 380 executing on higher level intermediate servers. The subprocess begins in step 502-1, in which the intermediate server responds to an attempt to establish communications by a higher level server. If the higher level server is master server 204, step 502-1 responds to an attempt by master server 204 to establish communications in step 418-2 of master distribution process 400. If the higher level server is a higher-level intermediate server, step 502-1 responds to an attempt by the higher level intermediate server to establish communications in step 518-2 of process 500. In either case, subprocess 502 responds to an attempt to establish communications by a higher level server.

In step 502-2, the subprocess receives data transmitted to the intermediate server from a higher level server in either step 418-3 or step 518-3. In step 502-3, the subprocess writes the received data to safe storage. In step 502-4, the subprocess sends an acknowledgment to the higher level server which receives it in either step 418-4 or step 518-4. The subprocess then ends.

Fig. 6a is a flow diagram of destination process 356, which is implemented by program instruction executed in the CPU of each destination server. It is best understood when viewed in conjunction with Fig. 2a. Process 356 begins in step 602, in which an update data record is received at a destination server from a higher level system. In step 604, the update data is written to the destination database. Process 356 then ends.

Fig. 6b is a flow diagram of a subprocess of step 602 of destination process 356. Subprocess 602 interacts with the subprocess of step 518 of instances of intermediate distribution process 380 executing on intermediate servers. Subprocess 602 is similar to the subprocess of step 502 of process 500, except that the transmitting server is always an intermediate server and the receiving server is a destination server.

Fig. 7 is an exemplary format of an update data record. Service Name 702 is the name of the service or data type being defined. This field is typically populated by the application creating the data. Sequence Number 704 is a number that allows sequences and ordering of service names. Distribution Queue 706 is the name of the messaging queue in which to place this data for delivery. Data Type 708 is an indicator of the representation type to use for the data. Formats such as, for example big endian, little endian, ASCII, EBCDIC, etc. are indicated. Module Name 710 is the name of a specialized data assembly module and/or data view module that is to be invoked for this distribution definition. Transaction Program Name 712 indicates the transaction program which is to be invoked as the message is being delivered. System ID 714 is the name of the remote system to which the data is to be distributed. Data 716 is the new data which is to be entered into the database.

Although specific embodiments have been disclosed, it is understood by those of skill in the art that other equivalent embodiments are possible.

CLAIMS

WHAT IS CLAIMED IS:

1. In a hierarchical distributed computing environment, including a plurality of servers arranged in at least three levels of hierarchy, the highest level of hierarchy including at least one highest-level server, the lowest level of hierarchy including at least one lowest-level server and there being at least one intermediate level of hierarchy including at least one intermediate-level server, each server including at least one database, a method for replicating changes to a database on the highest-level server comprising the steps of:

A) on the highest-level server, performing the steps of:

1) entering a change into a database stored on the highest-level server, and

2) transmitting an indicator of the change from the highest-level server to at least one server of an intermediate level of hierarchy;

B) on the intermediate-level server, performing the steps of:

1) receiving the indicator of the change from a server of higher level, which is either the highest-level server or a higher level intermediate-level server,

2) entering the change into the database of the intermediate-level server, and

3) transmitting an indicator of the change from the intermediate-level server to at least one lower level server, which is either a lowest-level server or a lower level intermediate-level server; and

C) on a lowest-level server, performing the steps of:

1) receiving an indicator of the change from an intermediate-level server, and

2) entering the change into the database of the lowest-level server.

2. The method of claim 1, wherein the highest-level server includes a distribution table and the highest-level server further performs the step of:

determining the intermediate-level server to which an indicator of the change is to be transmitted from the highest-level server by reference to a distribution table.

3. The method of claim 1, wherein the intermediate-level server includes a distribution table and the intermediate-level server further performs the step of:

determining the servers to which an indicator of the change is to be transmitted from the intermediate-level server by reference to a distribution table.

4. The method of claim 1, wherein the highest-level server further performs the step of:

reformatting the indicator of the change from the format of the highest-level server to the format of the intermediate-level server, the format being determined by reference to a distribution table.

5. The method of claim 1, wherein the intermediate-level server further performs the step of:

reformatting the indicator of the change from the format of the intermediate-level server to the format of the server to which the indicator is to be transmitted, the format being determined by reference to a distribution table.

6. The method of claim 1, wherein the highest-level server further performs the step of:

assembling the indicator of the change by use of data assembly routines indicated by a distribution table.

7. The method of claim 1, wherein the intermediate-level server further performs the step of:

assembling the indicator of the change by use of data assembly routines indicated by a distribution table.

8. The method of claim 1, wherein the highest-level server further performs the step of:

assembling the indicator of the change by use of data view routines indicated by a distribution table.

9. The method of claim 1, wherein the intermediate-level server further performs the step of:

assembling the indicator of the change by use of data view routines indicated by a distribution table.

10. The method of claim 1, wherein the step of the highest-level server transmitting the indicator of the change comprises the steps of:

A-2-1) repetitively transmitting the indicator of the change to at least one intermediate-level server, if previous transmissions were unsuccessful; and

A-2-2) transmitting an indicator of transmission failure, if the number of transmission failures exceeds a predetermined limit.

11. The method of claim 1, wherein the step of the intermediate-level server transmitting the indicator of the change comprises the steps of:

B-3-1) repetitively transmitting the indicator of the change to at least one lower level server, if previous transmissions were unsuccessful; and

B-3-2) transmitting an indicator of transmission failure, if the number of transmission failures exceeds a predetermined limit.

12. The method of claim 10, wherein the step of the highest-level server transmitting the indicator of the change further comprises the steps of:

A-2-3) saving the indicator of the change and an indicator of the progress of transmission of the indicator of the change; and

A-2-4) restoring the saved indicator of the change and an indicator of the progress of transmission of the indicator of the change, upon restoration of operation of the highest-level server after the highest-level server has failed.

13. The method of claim 11, wherein the step of the highest-level server transmitting the indicator of the change further comprises the steps of:

B-3-3) saving the indicator of the change and an indicator of the progress of transmission of the indicator of the change; and

B-3-4) restoring the saved indicator of the change and an indicator of the progress of transmission of the indicator of the change, upon restoration of operation of the highest-level server after the highest-level server has failed.

14. The method of claim 1, wherein each highest-level server and each intermediate-level server comprise a distribution table, data assembly routines indicated by the distribution table and an input queue which receives an indicator of the change, and wherein:

each highest-level server and each intermediate-level server, in response to the entry of the indicator of the change into the input queue performs the steps of:

determining the servers to which an indicator of the change is to be transmitted by reference to the distribution table;

reformatting the indicator of the change by reference to the distribution table; and assembling the indicator of the change by use of data assembly routines indicated by a distribution table.

15. The method of claim 14, wherein each highest-level server and each intermediate-level server comprises at least one communication queue which receives an indicator of the change after it has been reformatted and assembled, and wherein:

each server transmits the indicator of the change from the communication queue in response to receipt of the indicator by the communication queue.

16. The method of claim 14, wherein the highest-level server and the intermediate-level server further comprise:

one communication queue for each server to which an indicator of the change is to be transmitted.

17. A hierarchical distributed computing environment comprising:

A) a highest-level server transmitting an indicator of a change to a database to a lower level server;

B) an intermediate-level server, coupled to the highest level server, receiving the indicator of the change and transmitting it to a lower-level server;

C) a lowest-level server comprising a database, coupled to the intermediate-level server, receiving the indicator of the change and entering the change into the database; and

D) a data communications network, coupled to the servers, communicating data between the servers.

18. The hierarchical distributed computing environment of claim 17, wherein the highest-level server comprises:

A-1) a processor;

A-2) a memory, coupled to and accessible by the processor, storing data to be processed by the processor;

A-3) a database comprising data stored in the memory;

A-4) a data receiver, coupled to the processor, receiving an indicator of a change to be made to the data in the database;

A-5) a database modifier, coupled to the data receiver, entering modifications to the data in the database according to the received indicator of the change; and

A-6) a data transmitter, coupled to the processor and the data communications network, transmitting an indicator of the change from the highest-level server to at least one intermediate-level server.

19. The hierarchical distributed computing environment of claim 17, wherein the intermediate-level server comprises:

B-1) a processor;

B-2) a memory coupled to and accessible by the processor, storing data to be processed by the processor;

B-3) a database comprising data stored in the memory;

B-4) a data receiver. coupled to the processor and the data communications network, receiving an indicator of a change to be made to the data in the database;

B-5) a database modifier, coupled to the data receiver. entering modifications to the data in the database according to the received indicator of the change; and

B-6) a data transmitter. coupled to the processor and the data communications network. transmitting an indicator of the change from the from the intermediate-level server to at least one lower-level server. which is either a lowest-level server or a lower level intermediate-level server.

20. The hierarchical distributed computing environment of claim 17, wherein the lowest-level server comprises:

C-1) a processor;

C-2) a memory. coupled to and accessible by the processor. storing data to be processed by the processor;

C-3) a database comprising data stored in the memory;

C-4) a data receiver. coupled to the processor and the data communications network. receiving an indicator of a change to be made to the data in the database; and

C-5) a database modifier, coupled to the data receiver. entering modifications to the data in the database according to the received indicator of the change.

21. The hierarchical distributed computing environment of claim 18, wherein the highest-level server further comprises:

A-7) a distribution table stored in memory; and

A-8) a destination determiner coupled to the distribution table which determines the intermediate-level servers to which an indicator of the change is to be transmitted from the highest-level server by reference to the distribution table.

22. The hierarchical distributed computing environment of claim 19, wherein the intermediate-level server further comprises:

B-7) a distribution table stored in memory; and

B-8) a destination determiner coupled to the distribution table which determines the servers to which an indicator of the change is to be transmitted from the intermediate-level server by reference to the distribution table.

23. The hierarchical distributed computing environment of claim 18, wherein the highest-level server further comprises:

a reformatter coupled to the first data receiver which reformats the indicator of the change from the format of the highest-level server to the format of the intermediate-level server by reference to the distribution table.

24. The system of claim 19, wherein the intermediate-level server further comprises:

a reformatter coupled to the second data receiver which reformats the indicator of the change from the format of the intermediate-level server to the format of the server to which the indicator is to be transmitted by reference to a distribution table.

25. The hierarchical distributed computing environment of claim 18, wherein the highest-level server further comprises:

an indicator assembler which assembles the indicator of the change by use of data assembly routines indicated by a distribution table.

26. The hierarchical distributed computing environment of claim 19, wherein the intermediate-level server further comprises:

an indicator assembler which assembles the indicator of the change by use of data assembly routines indicated by a distribution table.

27. The hierarchical distributed computing environment of claim 18, wherein the highest-level server further comprises:

A-7) an input queue. contained in memory, receiving an indicator of the change: and

A-8) an indicator output device. coupled to the input queue, determining the servers to which the indicator is to be transmitted. reformatting the indicator and assembling the indicator in response to the entry of a change into the queue.

28. The hierarchical distributed computing environment of claim 19, wherein the intermediate-level server further comprises:

B-7) an input queue, contained in memory, receiving an indicator of the change; and

B-8) an indicator output device, coupled to the input queue, determining the servers to which the indicator is to be transmitted, reformatting the indicator and assembling the indicator in response to the entry of a change into the queue.

29. The hierarchical distributed computing environment of claim 18, wherein the data transmitter of the highest-level server comprises:

A-7) a data transmitter, coupled to the database modifier and at least one intermediate-level server, repetitively transmitting the indicator of the change to at least one intermediate-level server, if previous transmissions were unsuccessful; and

A-8) a failure transmitter, coupled to the repetitive data transmitter, transmitting an indicator of transmission failure, if the number of transmission failures exceeds a predetermined limit.

30. The hierarchical distributed computing environment of claim 19, wherein the data transmitter of the intermediate-level server comprises:

B-7) a data transmitter, coupled to the database modifier and at least one lower-level server, repetitively transmitting the indicator of the change to at least one lower level server, if previous transmissions were unsuccessful; and

B-8) a failure transmitter, coupled to the repetitive data transmitter, transmitting an indicator of transmission failure, if the number of transmission failures exceeds a predetermined limit.

31. The hierarchical distributed computing environment of claim 18, wherein the intermediate-level server comprises:

B-1) a processor;

B-2) a memory coupled to and accessible by the processor, storing data to be processed by the processor;

B-3) a database comprising data stored in the memory;

B-4) a data receiver, coupled to the processor and the data communications network, receiving an indicator of a change to be made to the data in the database;

B-5) a database modifier, coupled to the data receiver, entering modifications to the data in the database according to the received indicator of the change; and

B-6) a data transmitter, coupled to the processor and the data communications network, transmitting an indicator of the change from the from the intermediate-level server to at least one lower-level server, which is either a lowest-level server or a lower level intermediate-level server.

32. The system of claim 31, wherein each highest-level server and each intermediate-level server comprises:

a distribution table, stored in the memory;

data assembly routines, stored in the memory, indicated by the distribution table;

an input queue in the memory receiving an indicator of the change;

a destination determining device, coupled to the distribution table and the input queue, determining the servers to which an indicator of the change is to be transmitted by reference to the distribution table, in response to the entry of a change into the input queue;

a reformatter, coupled to the distribution table and the input queue, reformatting each indicator by reference to the distribution table, in response to the entry of a change into the input queue; and

an indicator assembler, coupled to the data assembly routines and the input queue, assembling the indicator of the change using data assembly routines indicated by a distribution table, in response to the entry of a change into the input queue.

33. The system of claim 32, wherein each highest-level server and each intermediate-level server further comprise:

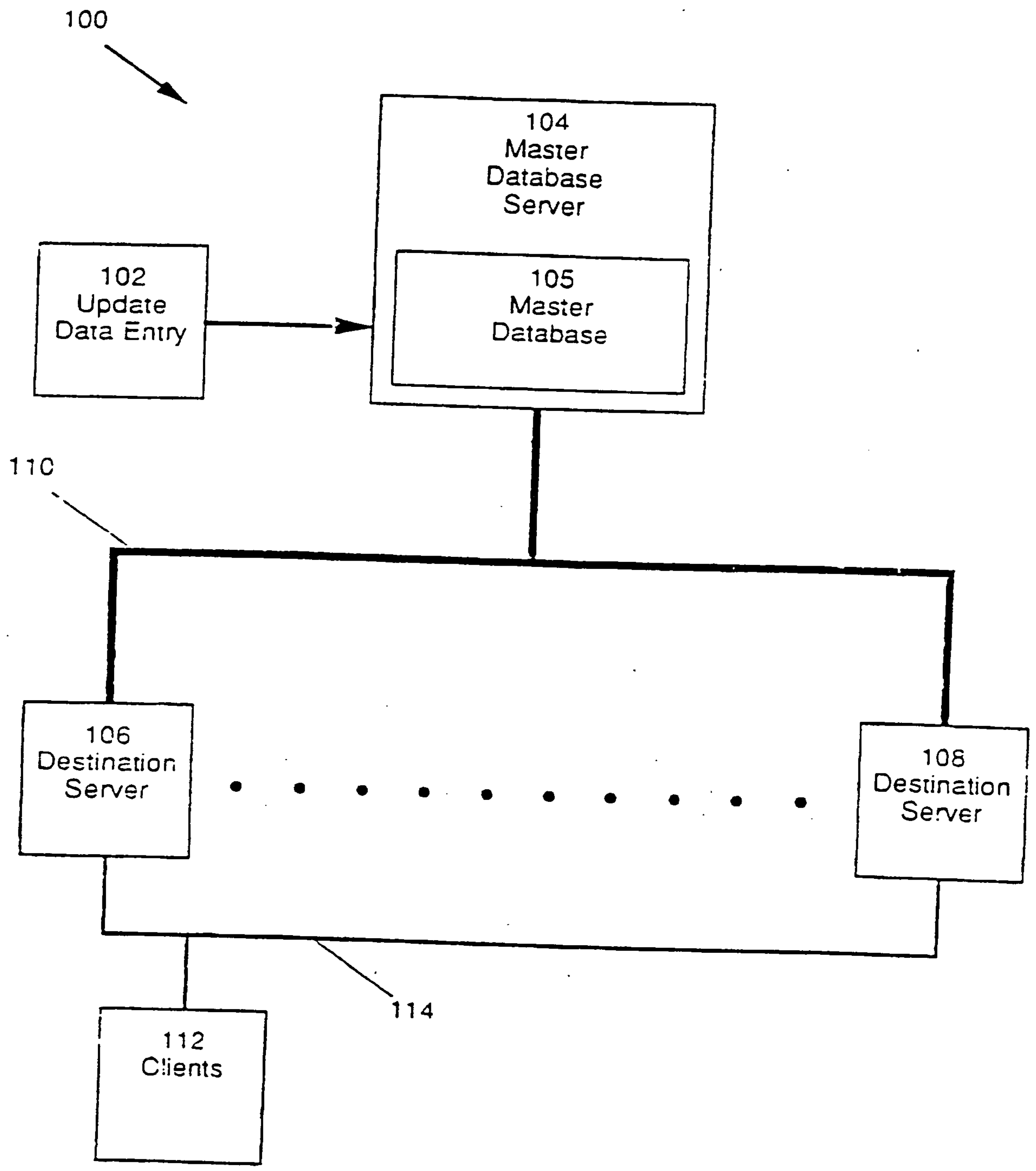
a communication queue, coupled to the indicator assembler, receiving an indicator of the change after it has been reformatted and assembled; and a data transmitter, coupled to the communication queue, transmitting the indicator of the change from the communication queue in response to receipt of the indicator by the communication queue.

34. The system of claim 33, wherein each highest-level server and each intermediate-level server further comprise:

one communication queue for each server to which an indicator of the change is to be transmitted.

FIG. 1

Prior Art



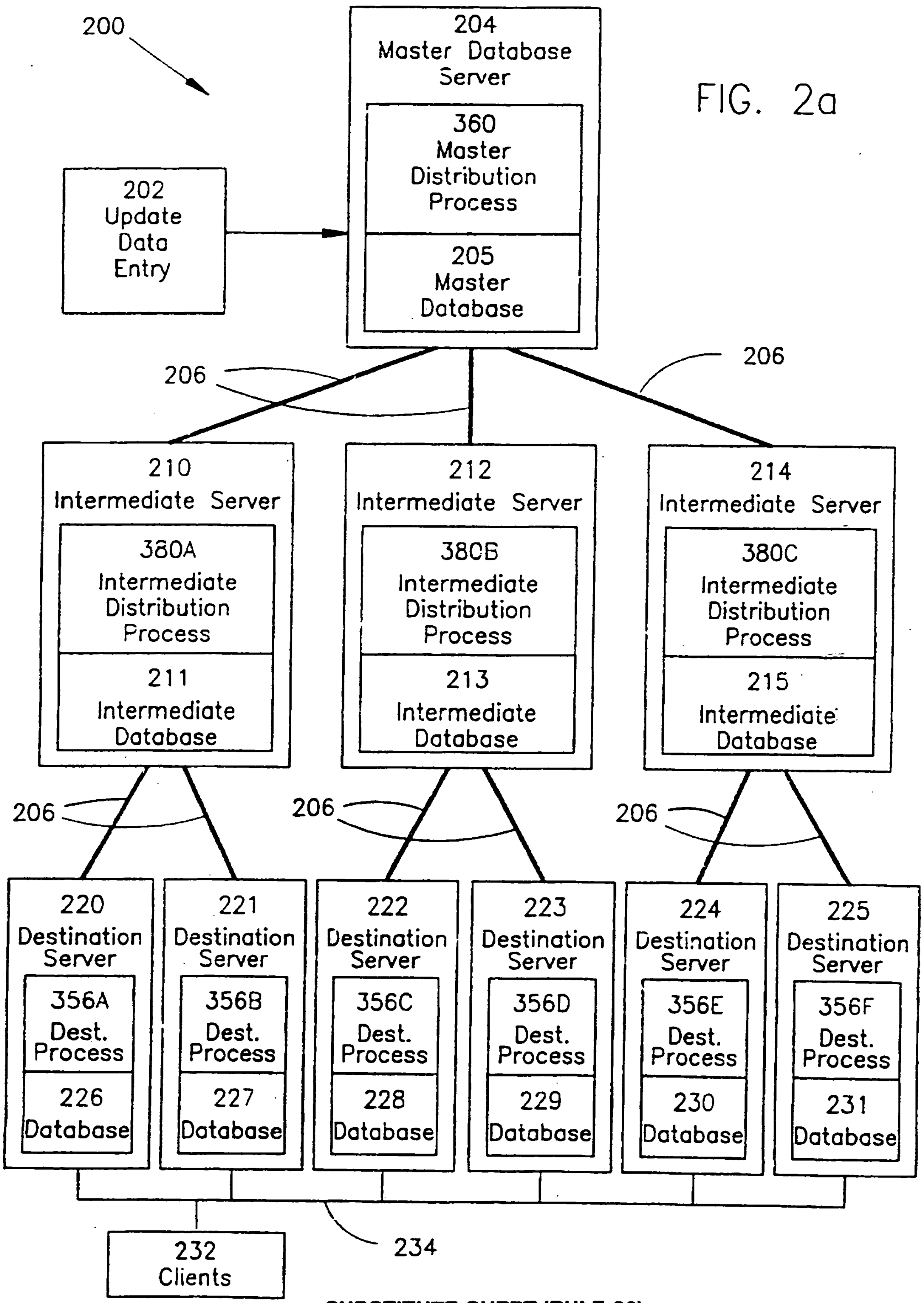

200
204 Master Database Server
360 Master Distribution Process
205 Master Database
202 Update Data Entry
206
210 Intermediate Server
380A Intermediate Distribution Process
211 Intermediate Database
212 Intermediate Server
380B Intermediate Distribution Process
213 Intermediate Database
214 Intermediate Server
380C Intermediate Distribution Process
215 Intermediate Database
220 Destination Server
356A Dest. Process
226 Database
221 Destination Server
356B Dest. Process
227 Database
222 Destination Server
356C Dest. Process
228 Database
223 Destination Server
356D Dest. Process
229 Database
224 Destination Server
356E Dest. Process
230 Database
225 Destination Server
356F Dest. Process
231 Database
232 Clients
234

FIG. 2a

FIG. 2b

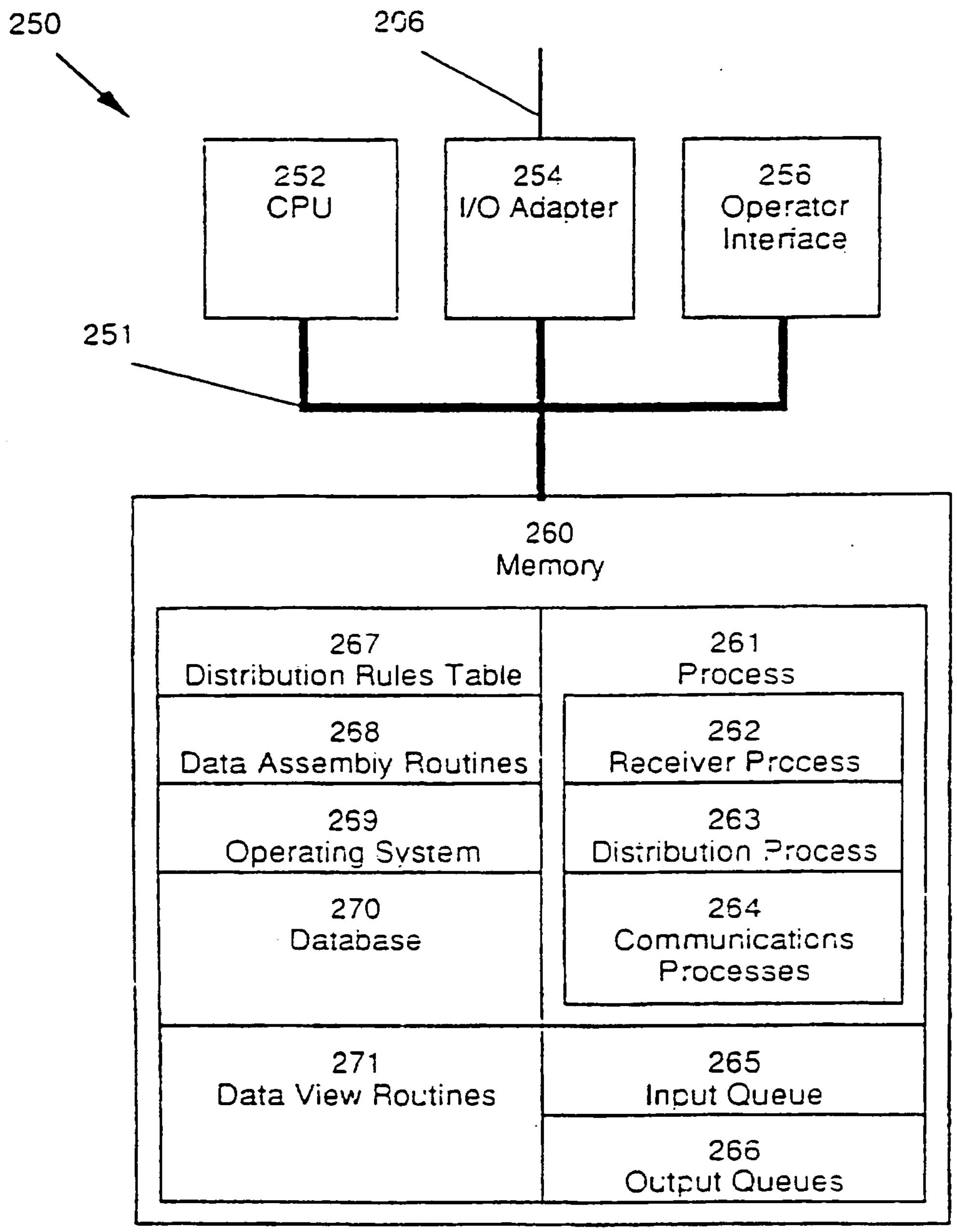

250
206
252 CPU
254 I/O Adapter
256 Operator Interface
251
260 Memory
267 Distribution Rules Table
268 Data Assembly Routines
269 Operating System
270 Database
261 Process
262 Receiver Process
263 Distribution Process
264 Communications Processes
271 Data View Routines
265 Input Queue
266 Output Queues

FIG. 2c

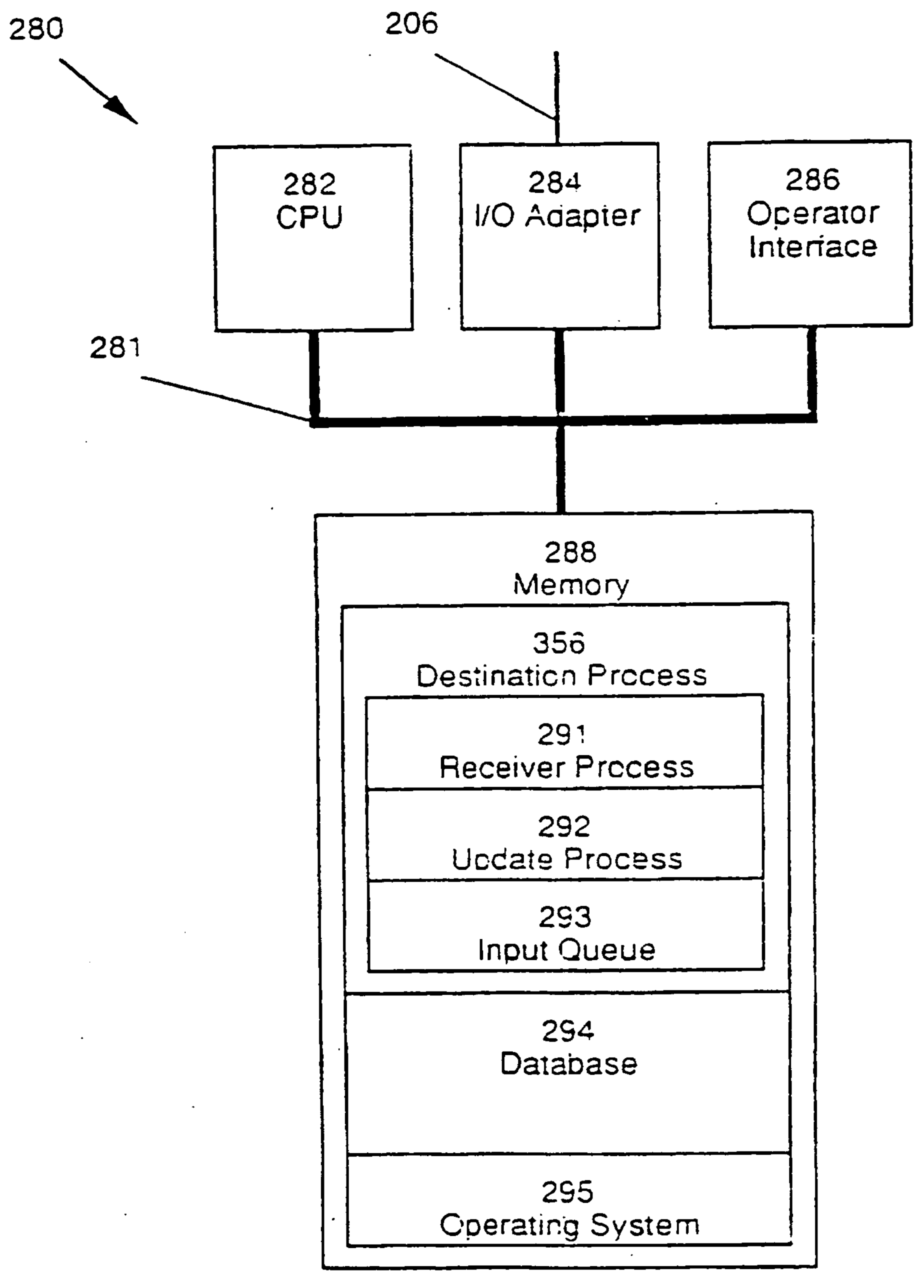

280
206
282
CPU
284
I/O Adapter
286
Operator Interface
281
288
Memory
356
Destination Process
291
Receiver Process
292
Update Process
293
Input Queue
294
Database
295
Operating System

FIG. 3a

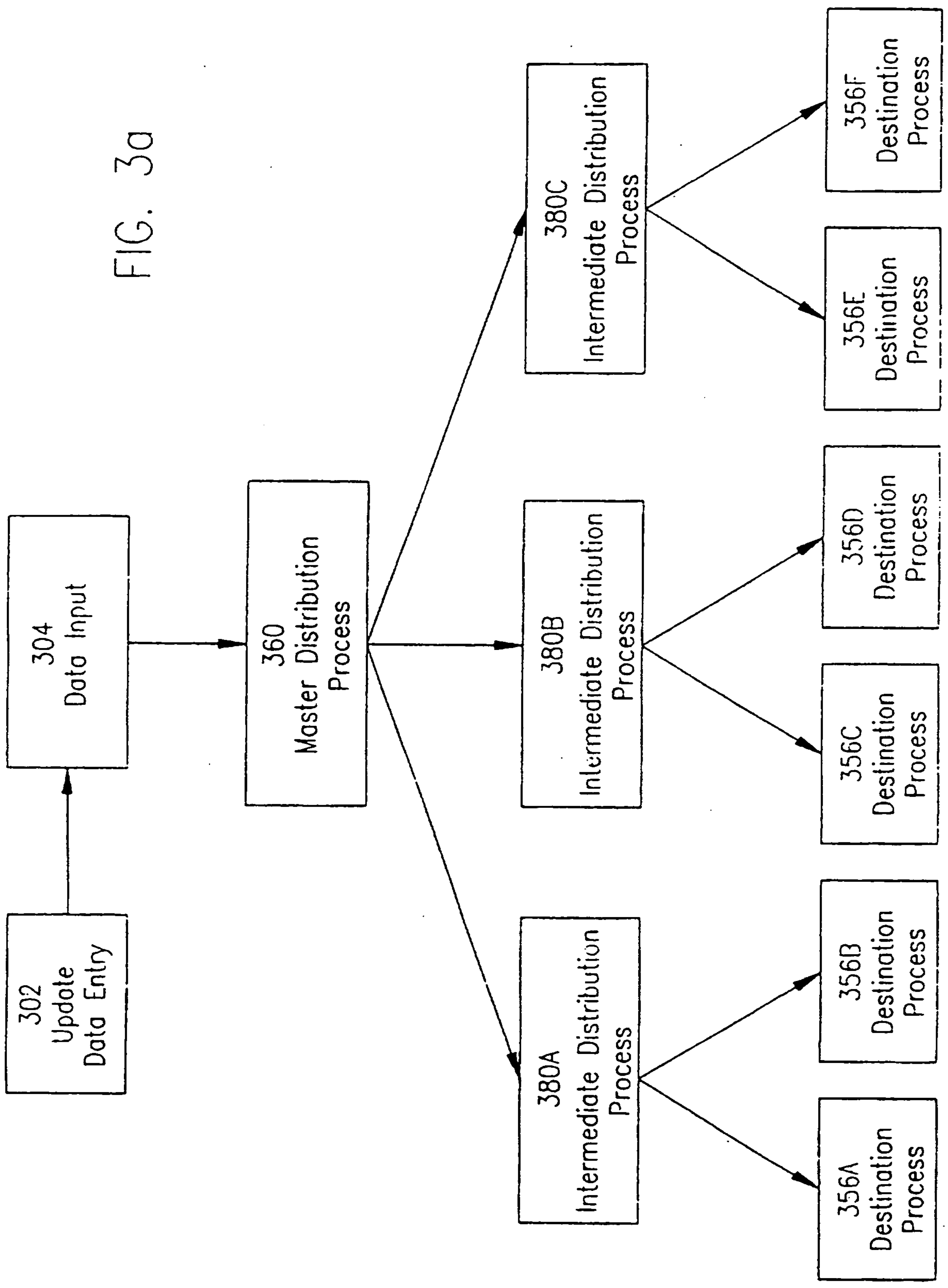
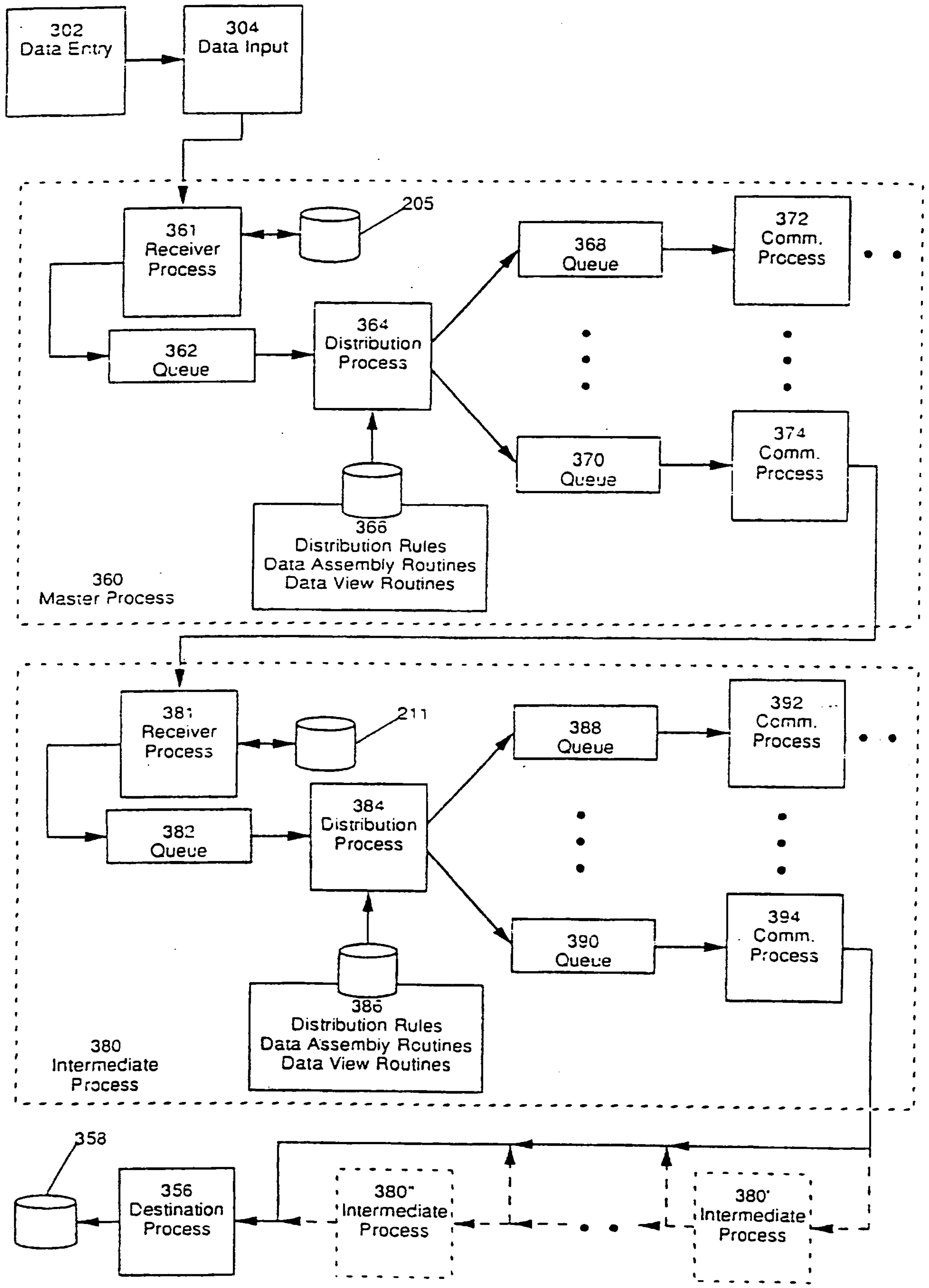

302
Update
Data Entry
304
Data Input
360
Master Distribution
Process
380A
Intermediate Distribution
Process
380B
Intermediate Distribution
Process
380C
Intermediate Distribution
Process
356A
Destination
Process
356B
Destination
Process
356C
Destination
Process
356D
Destination
Process
356E
Destination
Process
356F
Destination
Process

FIG. 3b


302 Data Entry
304 Data Input
361 Receiver Process
205
362 Queue
364 Distribution Process
366 Distribution Rules Data Assembly Routines Data View Routines
368 Queue
370 Queue
372 Comm. Process
374 Comm. Process
360 Master Process
381 Receiver Process
211
382 Queue
384 Distribution Process
386 Distribution Rules Data Assembly Routines Data View Routines
388 Queue
390 Queue
392 Comm. Process
394 Comm. Process
380 Intermediate Process
358
356 Destination Process
380" Intermediate Process
380' Intermediate Process

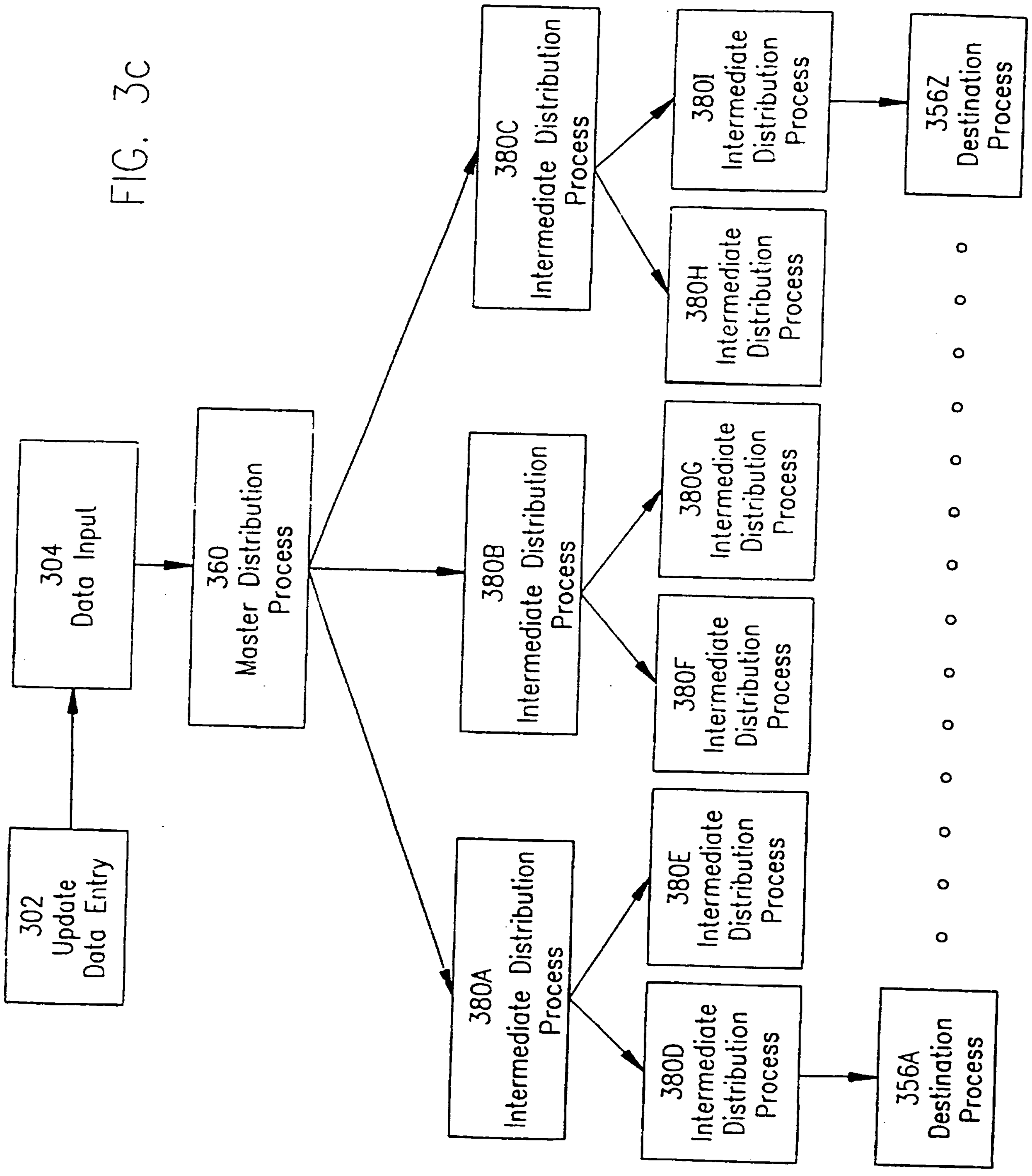

302 Update Data Entry
304 Data Input
360 Master Distribution Process
380A Intermediate Distribution Process
380B Intermediate Distribution Process
380C Intermediate Distribution Process
380D Intermediate Distribution Process
380E Intermediate Distribution Process
380F Intermediate Distribution Process
380G Intermediate Distribution Process
380H Intermediate Distribution Process
380I Intermediate Distribution Process
356A Destination Process
356Z Destination Process

FIG. 3C

FIG. 4a

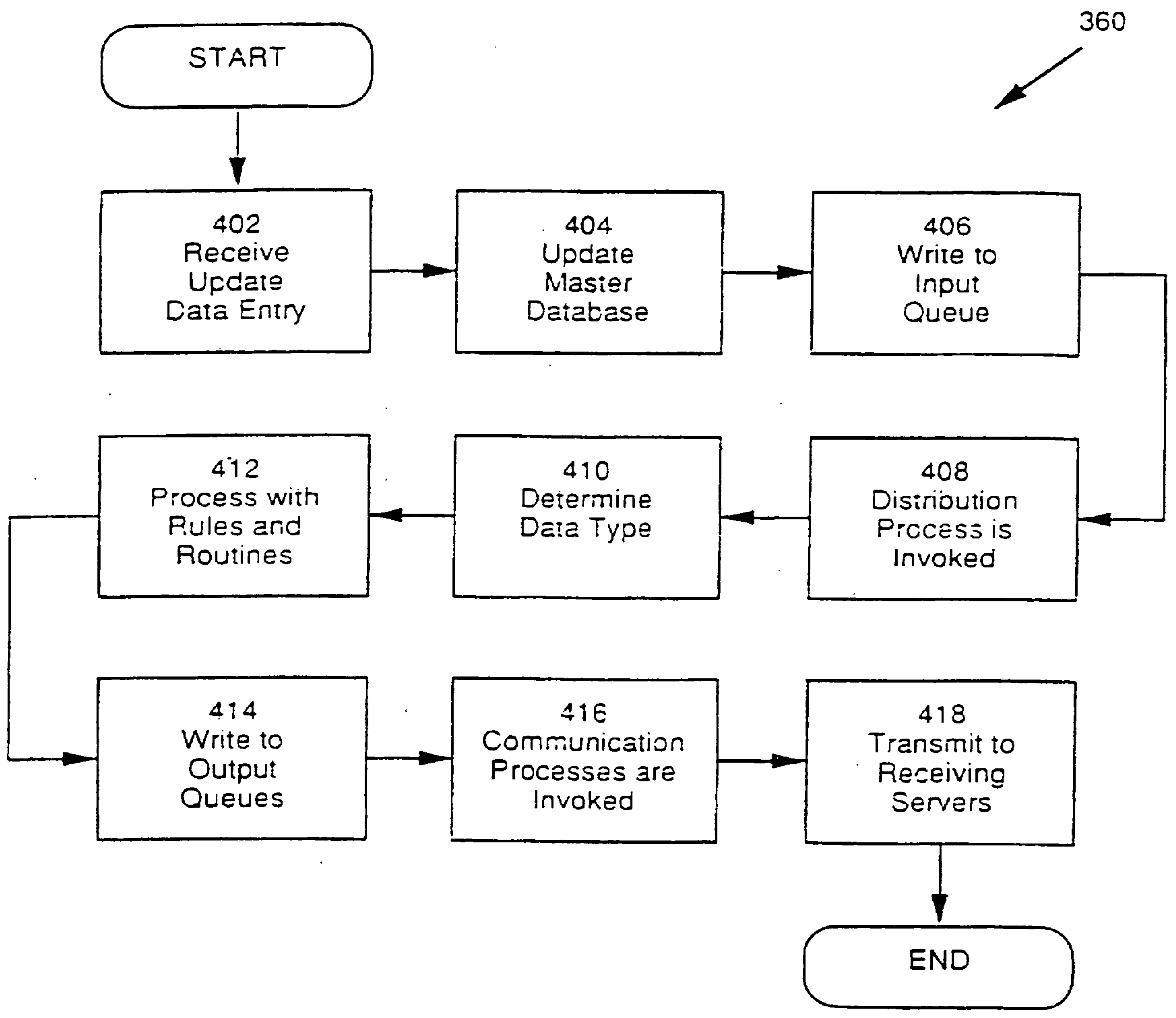
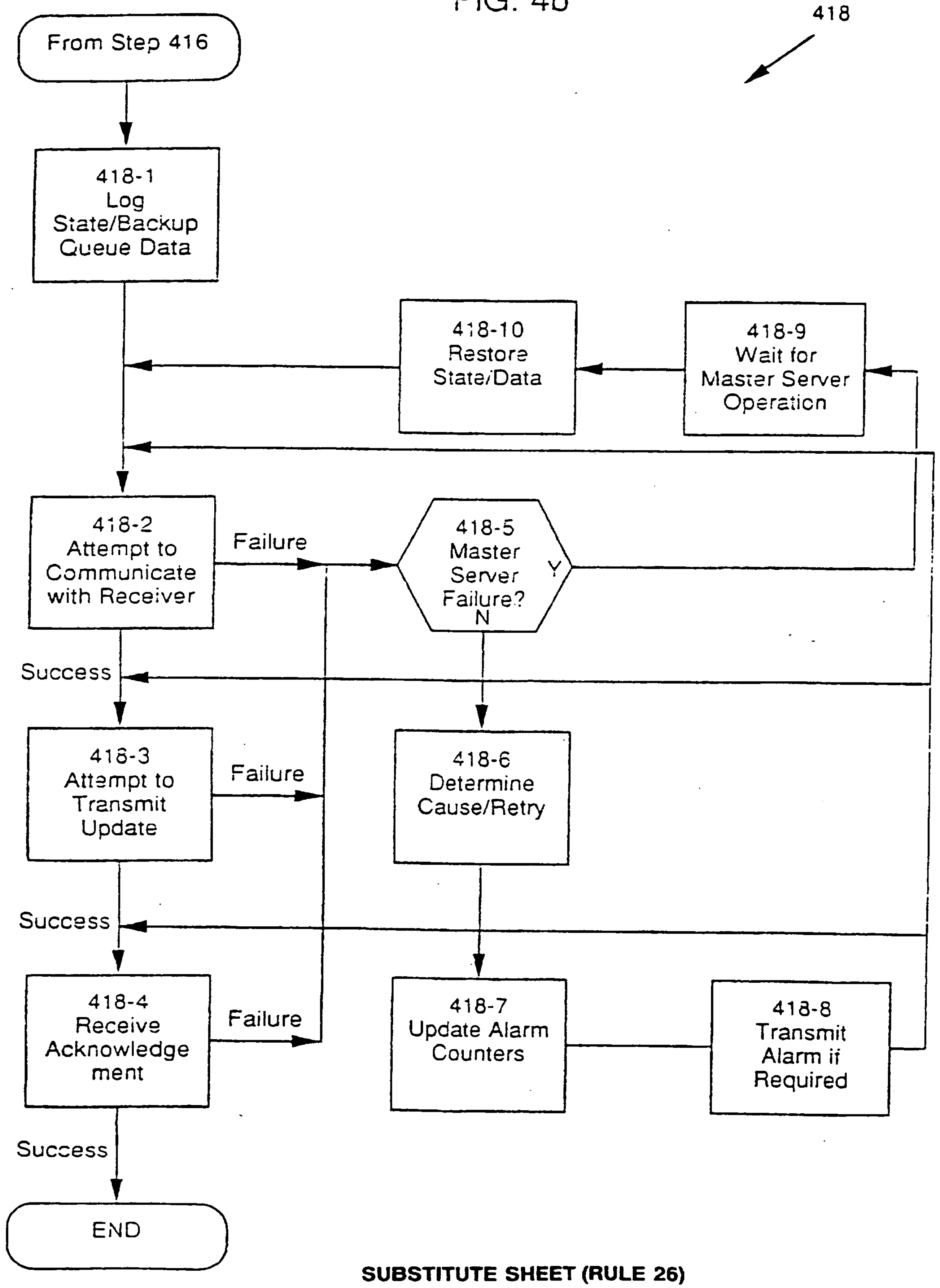

360
START
402 Receive Update Data Entry
404 Update Master Database
406 Write to Input Queue
408 Distribution Process is Invoked
410 Determine Data Type
412 Process with Rules and Routines
414 Write to Output Queues
416 Communication Processes are Invoked
418 Transmit to Receiving Servers
END

FIG. 4b


418
From Step 416
418-1 Log State/Backup Queue Data
418-10 Restore State/Data
418-9 Wait for Master Server Operation
418-2 Attempt to Communicate with Receiver
Failure
418-5 Master Server Failure?
Y
N
Success
418-3 Attempt to Transmit Update
Failure
418-6 Determine Cause/Retry
Success
418-4 Receive Acknowledge ment
Failure
418-7 Update Alarm Counters
418-8 Transmit Alarm if Required
Success
END

SUBSTITUTE SHEET (RULE 26)

FIG. 5a

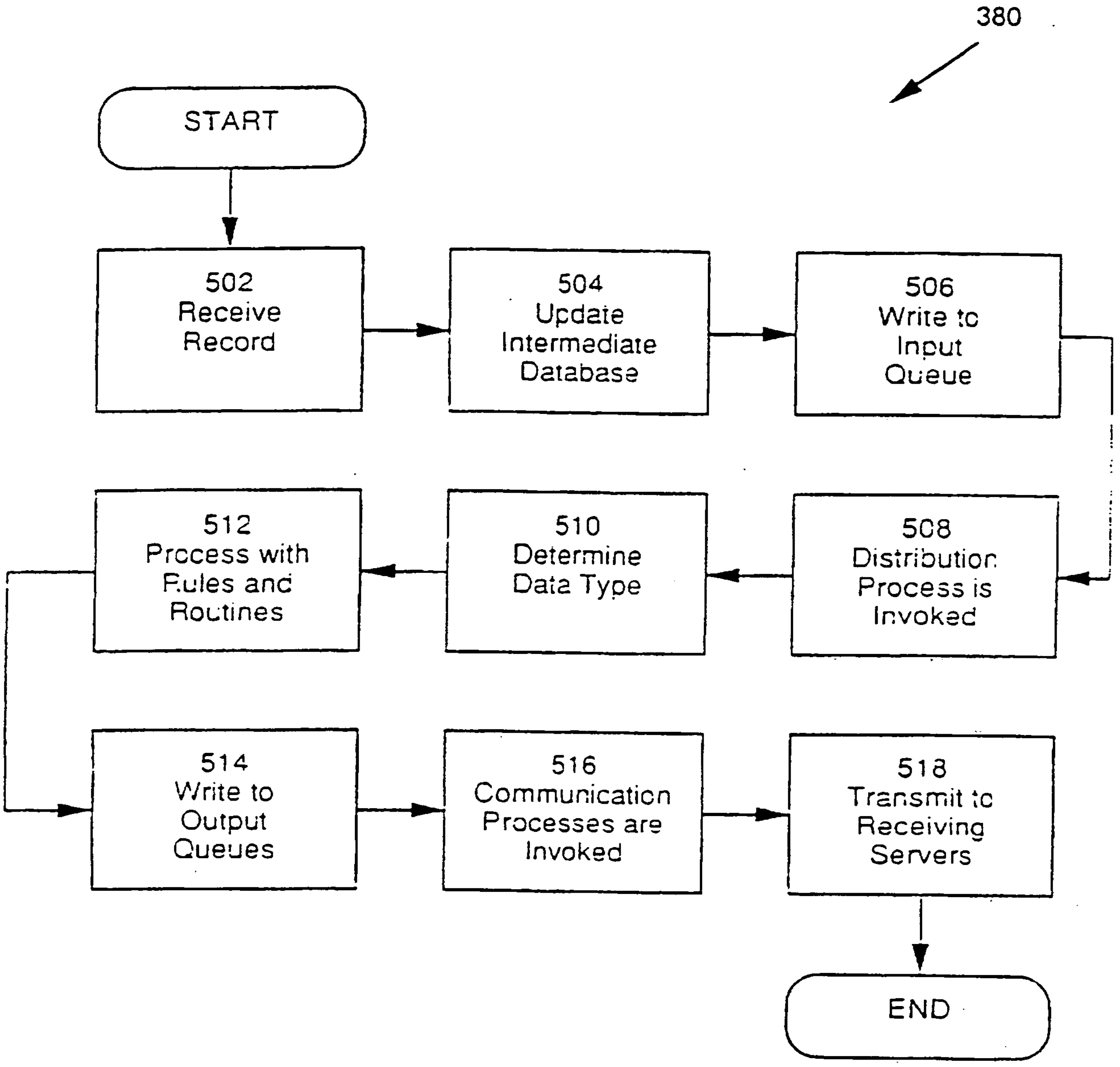

380
START
502 Receive Record
504 Update Intermediate Database
506 Write to Input Queue
508 Distribution Process is Invoked
510 Determine Data Type
512 Process with Rules and Routines
514 Write to Output Queues
516 Communication Processes are Invoked
518 Transmit to Receiving Servers
END

FIG. 5b

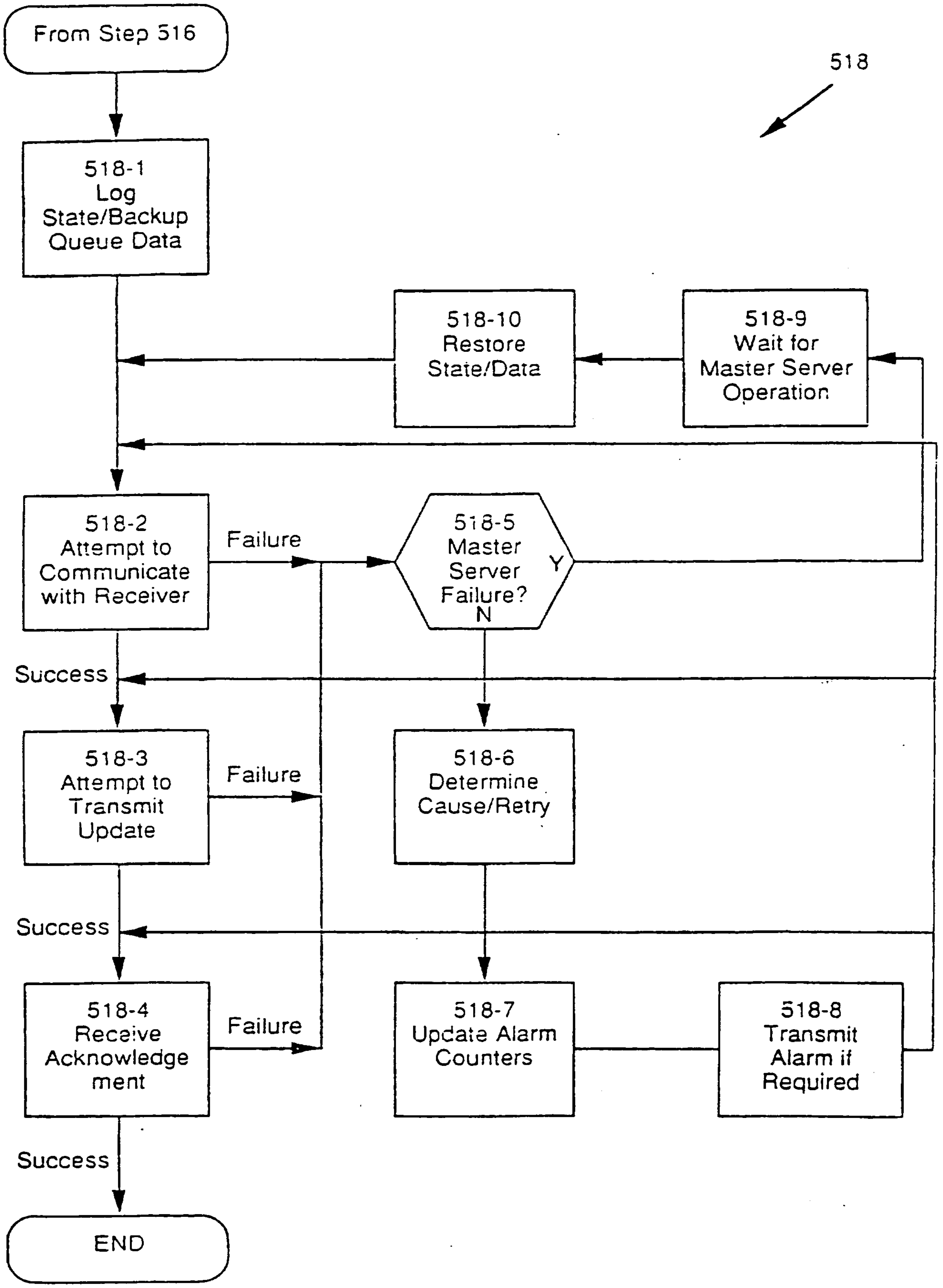

518
From Step 516
518-1 Log State/Backup Queue Data
518-10 Restore State/Data
518-9 Wait for Master Server Operation
518-2 Attempt to Communicate with Receiver
Failure
518-5 Master Server Failure?
Y
N
Success
518-3 Attempt to Transmit Update
Failure
518-6 Determine Cause/Retry
Success
518-4 Receive Acknowledge ment
Failure
518-7 Update Alarm Counters
518-8 Transmit Alarm if Required
Success
END

FIG. 5c

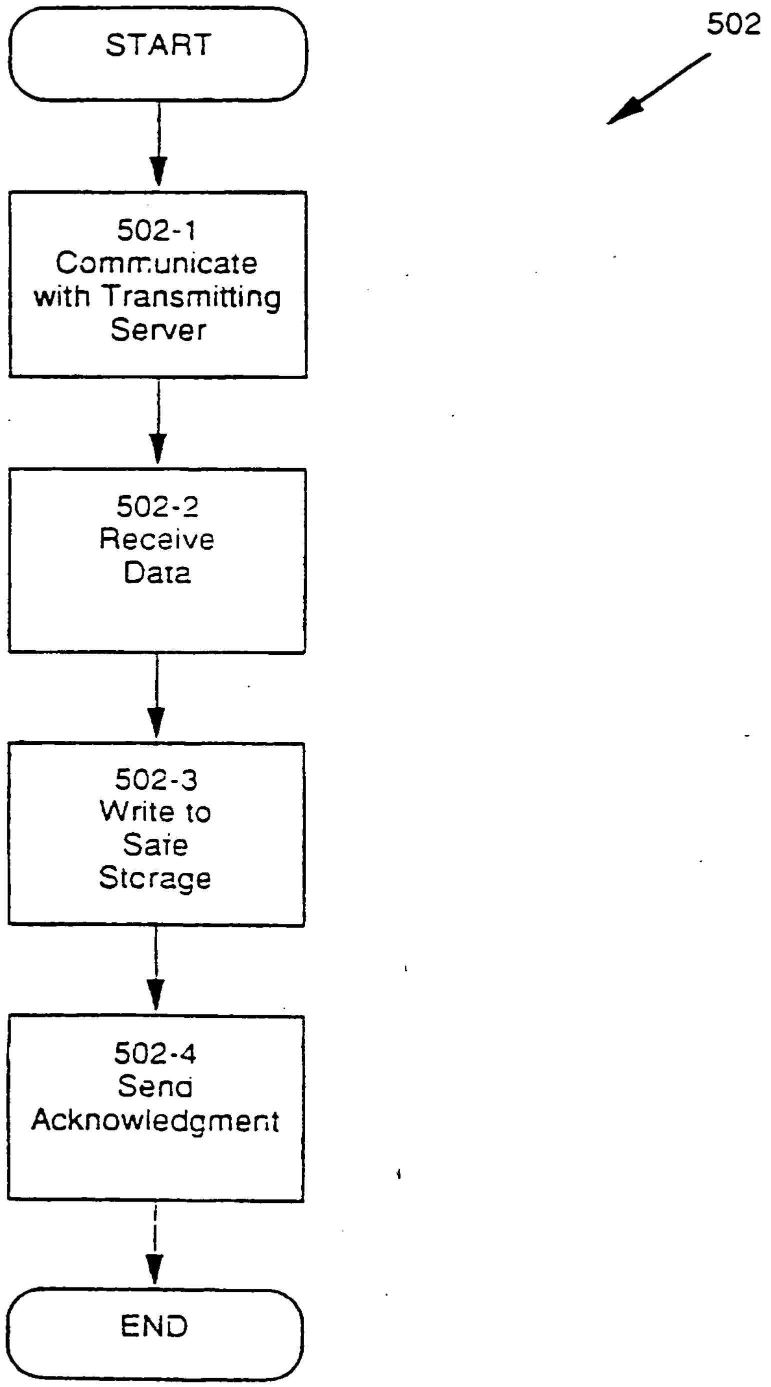
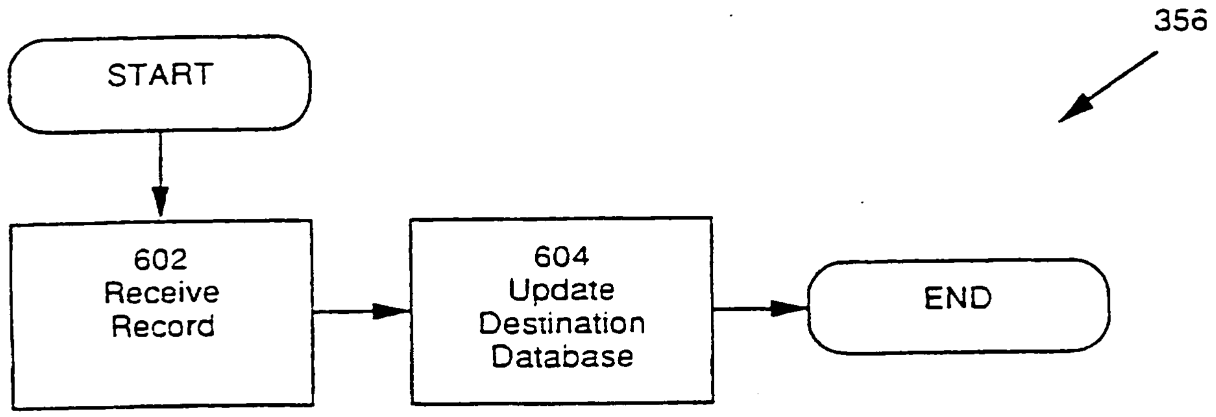

START
502
502-1 Communicate with Transmitting Server
502-2 Receive Data
502-3 Write to Safe Storage
502-4 Send Acknowledgment
END

FIG. 6a



356
START
602
Receive
Record
604
Update
Destination
Database
END

FIG. 6b

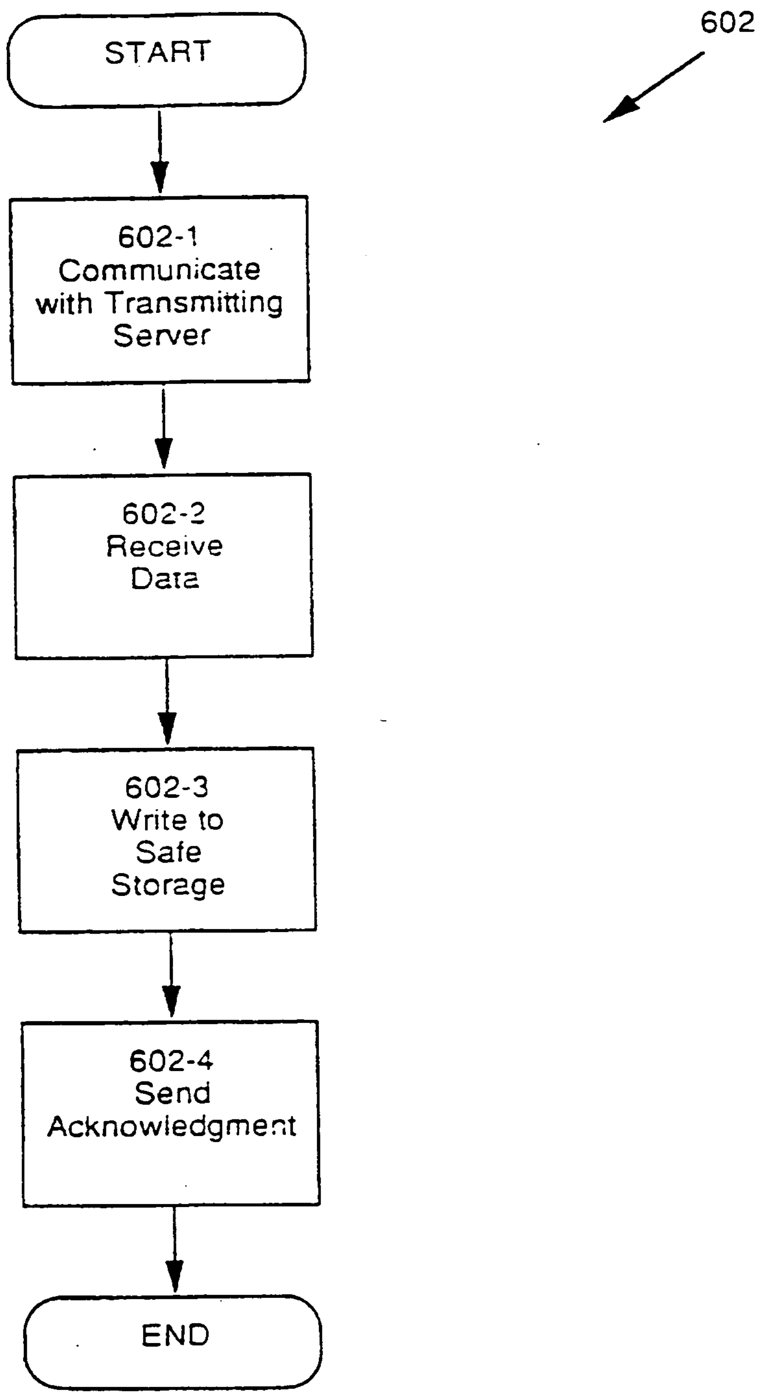
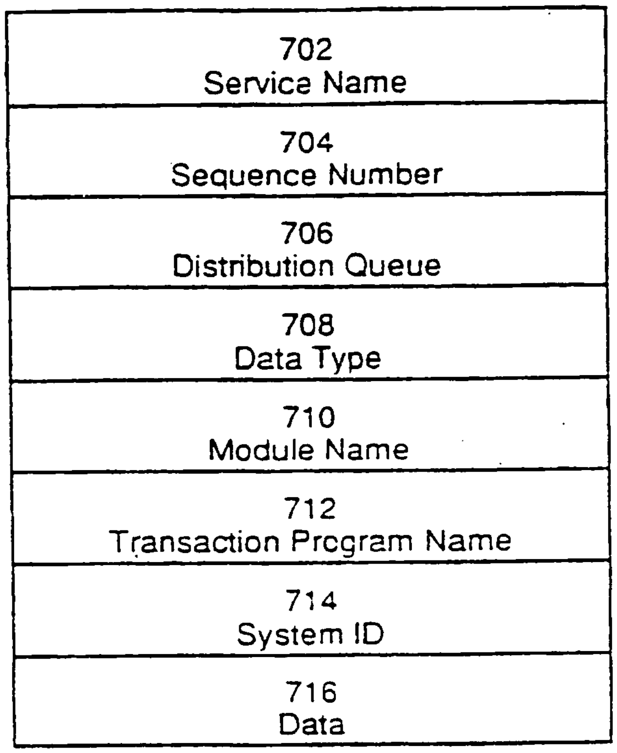

602
START
602-1
Communicate with Transmitting Server
602-2
Receive Data
602-3
Write to Safe Storage
602-4
Send Acknowledgment
END

FIG. 7


702
Service Name
704
Sequence Number
706
Distribution Queue
708
Data Type
710
Module Name
712
Transaction Program Name
714
System ID
716
Data

INTERNATIONAL SEARCH REPORT	International application No. PCT/US96/14994

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/30
US CL :364/200.01, 395/610, 621, 681
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/200.01, 395/610, 621, 681,712,800

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	USA 5,006,978 (NECHES) 09 April 1991, col.4, lines 27-68	1-34
Y	USA 5,276,899 (NECHES) 04 January 1994, col.4, lines 27-68	
Y	USA 5,530,855 (SATOH et al.) 25 June 1996, col.3, lines 16-68	
Y	USA 5,537,585 (BLICKENSTAFF et al.) 16 July 1996, col.2, lines 9-68	

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 DECEMBER 1996	13 FEB 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer [illegible] Thomas Black
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9707